Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 1er. Semestre 2021.

Implementación de un API que brinda servicios utilizando el protocolo HTTP bajo el concepto de (POO) y el uso de bases de datos.

Carnet 201904013 – Marlon Isaí Figueroa Farfán

Resumen

Para llevar a cabo el desarrollo de una solución integral que implemente un API que brinde servicios utilizando el protocolo HTTP bajo el concepto de (POO) y el uso de bases de datos se empezó definiendo el concepto de protocolo HTTP y las ventajas de implementarlo para consumir una API, planteando los diferentes modelos y diseños convenientes para desarrollar una arquitectura de aplicación óptima y manejar de la mejor forma posible los servicios proporcionados por el servidor al cliente. Por otro lado, se aclararon las ventajas y como trabajar con los archivos XML que sirvieron de insumos para la comunicación con el API desarrollado.

Palabras clave

1. ***Servidor:*** sistema que proporciona recursos, datos, servicios o programas a otros ordenadores.
2. ***Protocolo HTTP:*** protocolo de comunicación que permite las transferencias de información.
3. ***POO:*** Programación Orientada a objetos.
4. ***API:*** interfaz de programación de aplicaciones.
5. ***REGEX:*** se utiliza para describir o encontrar patrones dentro de una cadena.

Abstract

To carry out the development of an integral solution that implements an API that provides services using the HTTP protocol under the concept of (POO) and the use of databases, we began by defining the concept of HTTP protocol and the advantages of implementing it to consume an API, proposing the different models and convenient designs to develop an optimal application architecture and manage in the best possible way the services provided by the server to the client. On the other hand, the advantages and how to work with the XML files that served as inputs for the communication with the developed API were clarified.

Keywords

1. ***Server: system that provides resources, data, services or programs to other computers.***
2. ***HTTP protocol: communication protocol that allows information transfers.***
3. ***OOP: Object Oriented Programming.***
4. ***API: application programming interface.***
5. ***REGEX: used to describe or find patterns within a string.***

INTRODUCCIÓN

En la actualidad existen diferentes modelos y diseños para construir una buena arquitectura de software de las cuales podemos resaltar el modelo de cliente-servidor que es uno de los estilos arquitectónicos distribuidos más conocidos, el cual está compuesto por dos componentes, el proveedor y el consumidor. El proveedor es un servidor que brinda una serie de servicios o recursos los cuales son consumido por el Cliente.

Desarrollo del tema

I. ¿QUÉ ES UN SERVIDOR WEB?

La principal función de un servidor Web es almacenar los archivos de un sitio y emitirlos por Internet para poder ser visitado por los usuarios. Básicamente, un servidor Web es una gran computadora que guarda y transmite datos vía el sistema de redes llamado Internet. Cuando un usuario entra en una página de Internet, su navegador se comunica con el servidor enviando y recibiendo datos que determinan qué es lo que ve en la pantalla. Por eso, decimos que los servidores Web están para almacenar y transmitir datos de un sitio según lo solicita el navegador de un visitante.

II. PROGRAMACIÓN ORIENTADA A OBJETOS

Para definir la funcionalidad de la programación orientada a objetos se debe mencionar que ésta se compone de:

Objeto: Permite separar los diferentes componentes de un programa, simplificando su elaboración.

A los objetos se les otorga ciertas características en la vida real. Cada parte del programa que se desea realizar es tratado como objeto.

Clase: Consta de una serie de métodos y datos que resumen las características de este objeto. Definir clases permite trabajar con código reutilizable.

Atributos: Tiene unas características que le identifican como tal, así:

OBJETO	CARACTERÍSTICAS
rectángulo	lados
	ángulos
persona	nombre
	cédula
	fecha Nacimiento
libro	título
	autor
	año

Figura 1. Ejemplo de Atributo

Fuente: elaboración propia

Métodos: Sobre los objetos es posible realizar acciones (operaciones, cálculos), para lo cual podría utilizarse la pregunta, ¿Qué se puede hacer sobre el objeto X? Las respuestas a esta interrogante facilitarán la identificación de las funcionalidades de los objetos.

OBJETO	FUNCIONALIDADES
rectángulo	CalcularArea
	CalcularPerimetro
detalleFactura	CalcularIVA
	CalcularDescuento
	CalcularTotal
transacción	Depositar
	Retirar
	CalcularIntereses

Figura II. Ejemplo de Métodos

Fuente: Elaboración propia

III. PROTOCOLO HTTP

HTTP, de sus siglas en inglés: "Hypertext Transfer Protocol", es el nombre de un protocolo el cual nos permite realizar una petición de datos y recursos, como pueden ser documentos HTML. Es la base de cualquier intercambio de datos en la Web, y un protocolo de estructura cliente-servidor, esto quiere decir que una petición de datos es iniciada por el elemento que recibirá los datos (el cliente), normalmente un navegador Web. Así, una página web completa resulta de la unión de distintos subdocumentos recibidos, como, por ejemplo: un documento que especifique el estilo de maquetación de la página web (CSS), el texto, las imágenes, vídeos, scripts, etc...

CARACTERÍSTICAS CLAVE DEL PROTOCOLO HTTP

HTTP ES SENCILLO

Incluso con el incremento de complejidad, que se produjo en el desarrollo de la versión del protocolo HTTP/2, en la que se encapsularon los mensajes, HTTP está pensado y desarrollado para ser leído y fácilmente interpretado por las personas, haciendo de esta manera más fácil la depuración de errores, y reduciendo la curva de aprendizaje para las personan que empieza a trabajar con él.

HTTP ES UN PROTOCOLO CON SESIONES, PERO SIN ESTADOS

HTTP es un protocolo sin estado, es decir: no guarda ningún dato entre dos peticiones en la misma sesión. Esto crea problemáticas, en caso de que los usuarios requieran interactuar con determinadas páginas Web de forma ordenada y coherente, por ejemplo, para el uso de "cestas de la compra" en páginas que utilizan en comercio electrónico. Pero, mientras HTTP ciertamente es un protocolo sin estado, el uso de HTTP cookies, si permite guardar datos con respecto a la sesión de comunicación. Usando la capacidad de ampliación del protocolo HTTP, las cookies permiten crear un contexto común para cada sesión de comunicación.

PETICIONES

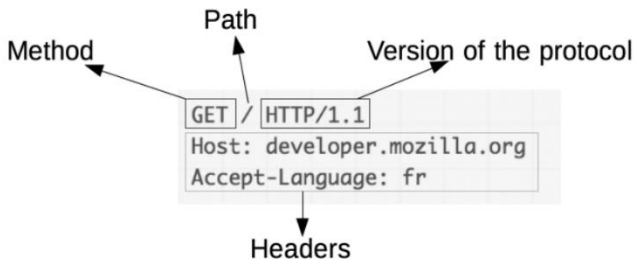


Figura III. Ejemplo de petición

Fuente: elaboración propia

Una petición de HTTP, está formado por los siguientes campos:

- Un método HTTP, normalmente pueden ser un verbo, como: GET, POST o un nombre como: OPTIONS (en-US) o HEAD (en-US), que defina la operación que el cliente quiera realizar. El objetivo de un cliente, suele ser una petición de recursos, usando GET, o presentar un valor de un formulario HTML, usando POST, aunque en otras ocasiones puede hacer otros tipos de peticiones.
- La dirección del recurso pedido; la URL del recurso, sin los elementos obvios por el contexto, como pueden ser: sin el protocolo (http://), el dominio (aquí developer.mozilla.org), o el puerto TCP (aquí el 80).
- La versión del protocolo HTTP.
- Cabeceras HTTP opcionales, que pueden aportar información adicional a los servidores.
- un cuerpo de mensaje, en algún método, como puede ser POST, en el cual envía la información para el servidor.

RESPUESTA

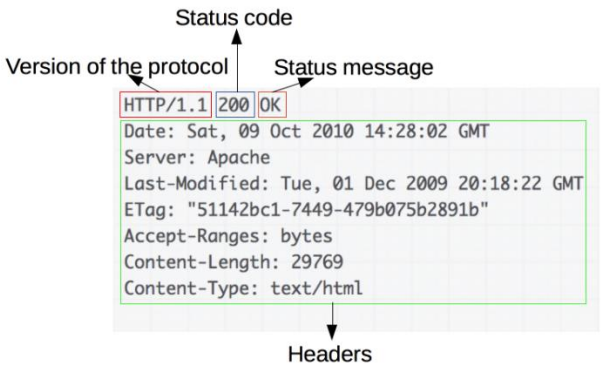


Figura 1V. Ejemplo de respuesta

Fuente: elaboración propia

IV. API

El término API es una abreviatura de **Application Programming Interfaces**, que en español significa interfaz de programación de aplicaciones. Se trata de un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones, permitiendo la comunicación entre dos aplicaciones de software a través de un conjunto de reglas.

Así pues, podemos hablar de una API como una especificación formal que establece cómo un módulo de un software se comunica o interactúa con otro para cumplir una o muchas funciones. Todo dependiendo de las aplicaciones que las vayan a utilizar, y de los permisos que les dé el propietario de la API a los desarrolladores de terceros.

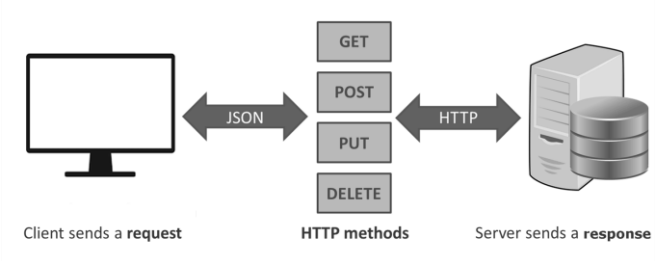


Figura 1V. Arquitectura API

Fuente: elaboración propia

¿PARA QUÉ SIRVE UNA API?

Una API sirve para conectar código o funciones entre plataformas diferentes, para que se puedan aprovechar los servicios de una web en otra. Su uso cobra especial importancia dentro del terreno de las redes sociales, como ya hemos indicado, debido a que posibilita la interacción directa con estas desde páginas web.

Funciona cuando se desea una publicación automática en lugares como Facebook o Twitter al lanzar un nuevo contenido, para que un CRM pueda conectarse con la app de una compañía y para mil tareas más que requieran de esa interacción entre plataformas y formatos diferentes. Sirve como nexo entre herramientas y sistemas con el fin de agilizar tareas y reducir la carga de trabajo.

V. MODELOS Y DISEÑOS DE ARQUITECTURA

Para utilizar la arquitectura de software se sigue un conjunto de patrones arquitectónicos, entre los cuales podemos encontrar:

- Cliente-Servidor
- Blackboard.
- Modelo entre capas.
- Intérprete.
- Orientado a servicios

La arquitectura de software cuenta con varios modelos, ellos son:

MODELOS ESTRUCTURALES

Son similares a la vista estructural, pero su énfasis primario radica en la (usualmente una sola) estructura coherente del sistema completo, en vez de concentrarse en su composición. Los modelos de framework a menudo se refieren a dominios o clases de problemas específicos. El trabajo que ejemplifica esta variante incluye arquitecturas de software específicas de dominios, como CORBA, o modelos basados en CORBA, o repositorios de componentes específicos, como PRISM.

MODELOS DINÁMICOS

Enfatizan la cualidad conductual de los sistemas, “Dinámico” puede referirse a los cambios en la configuración del sistema, o a la dinámica involucrada en el progreso de la computación, tales como valores cambiantes de datos.

MODELOS DE PROCESO

Se concentran en la construcción de la arquitectura, y en los pasos o procesos involucrados en esa construcción. En esta perspectiva, la arquitectura es el resultado de seguir un argumento (script) de proceso. Esta vista se ejemplifica con el actual trabajo sobre programación de procesos para derivar arquitecturas.

VI. ARQUITECTURA CLIENTE-SERVIDOR

Cliente-Servidor es uno de los estilos arquitectónicos distribuidos más conocidos, el cual está compuesto por dos componentes, el proveedor y el consumidor. El proveedor es un servidor que brinda una serie de servicios o recursos los cuales son consumido por el Cliente.

En una arquitectura Cliente-Servidor existe un servidor y múltiples clientes que se conectan al servidor para recuperar todos los recursos necesarios para funcionar, en este sentido, el cliente solo es una capa para representar los datos y se detonan acciones para modificar el estado del servidor, mientras que el servidor es el que hace todo el trabajo pesado.

Como podemos ver en la imagen de abajo, el cliente y el servidor son construidos en lo general como Monolíticos, donde cada desarrollo crea su propio ejecutable único y funciona sobre un solo equipo, con la diferencia de que estas aplicaciones no son autosuficientes, ya que existe una dependencia simbiótica entre los dos.

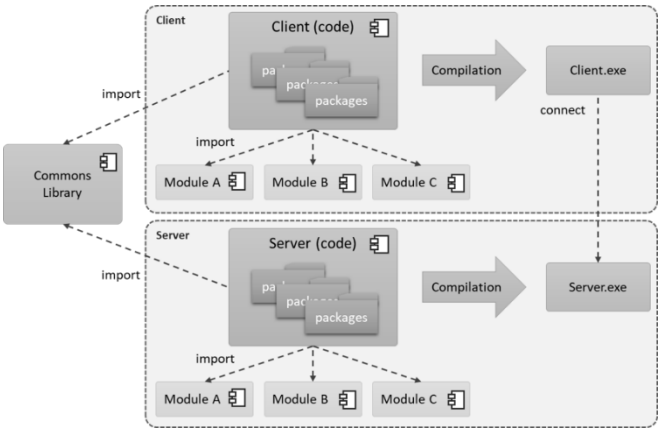


Figura III. Ejemplo de estructura cliente-servidor

Fuente: elaboración propia

VENTAJAS

- **Centralización:** El servidor fungirá como única fuente de la verdad, lo que impide que los clientes conserven información desactualizada.
- **Seguridad:** El servidor por lo general está protegido por firewall o subredes que impiden que los atacantes pueden acceder a la base de datos o los recursos sin pasar por el servidor.
- **Fácil de instalar (cliente):** El cliente es por lo general una aplicación simple que no tiene dependencias, por lo que es muy fácil de instalar.
- S**eparación de responsabilidade**s: La arquitectura cliente-servidor permite implementar la lógica de negocio de forma separada del cliente.
- **Portabilidad:** Una de las ventajas de tener dos aplicaciones es que podemos desarrollar cada parte para correr en diferentes plataformas, por ejemplo, el servidor solo en Linux, mientras que el cliente podría ser multiplataforma.

VII. ¿QUÉ ES UN ARCHIVO?

Para poder acceder a determinada información en cualquier momento, se necesitará que ella esté almacenada en forma permanente. Este es el caso de la memoria externa o auxiliar en las cuales la información permanece allí.

La forma de guardar los datos en estos dispositivos auxiliares es mediante unas estructuras llamadas **archivos** o ficheros.

ARCHIVO XML

XML son las siglas de ***Extensible Markup Language***, los archivos XML contienen información de cualquier tipo, se componen de etiquetas que nos aportan datos e información que queremos procesar. Estas etiquetas pueden estar de forma **individual o anidadas**.

ESTRUCTURA

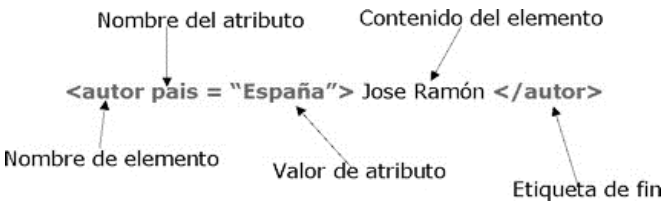


Figura V. Estructura de archivo XML

Fuente: Elaboración propia

Conclusiones

- La implementación de POO facilita el desarrollo de la solución para proyectos.
- La utilización de estructuras de programación secuenciales, cíclicas y condicionales permiten un mejor desarrollo y optimización.
- Para una mejor visualización de la correcta manipulación de datos con TDA's se utilizó un software externo (Graphviz).
- La utilización de archivos XML como insumos para la lógica y comportamiento contribuye a una óptima resolución.

Referencias bibliográficas

Desarrolloweb.com. 2004. *Usabilidad y arquitectura del software*. [online] Available at: <https://desarrolloweb.com/articulos/1622.php> [Accessed 3 May 2021].

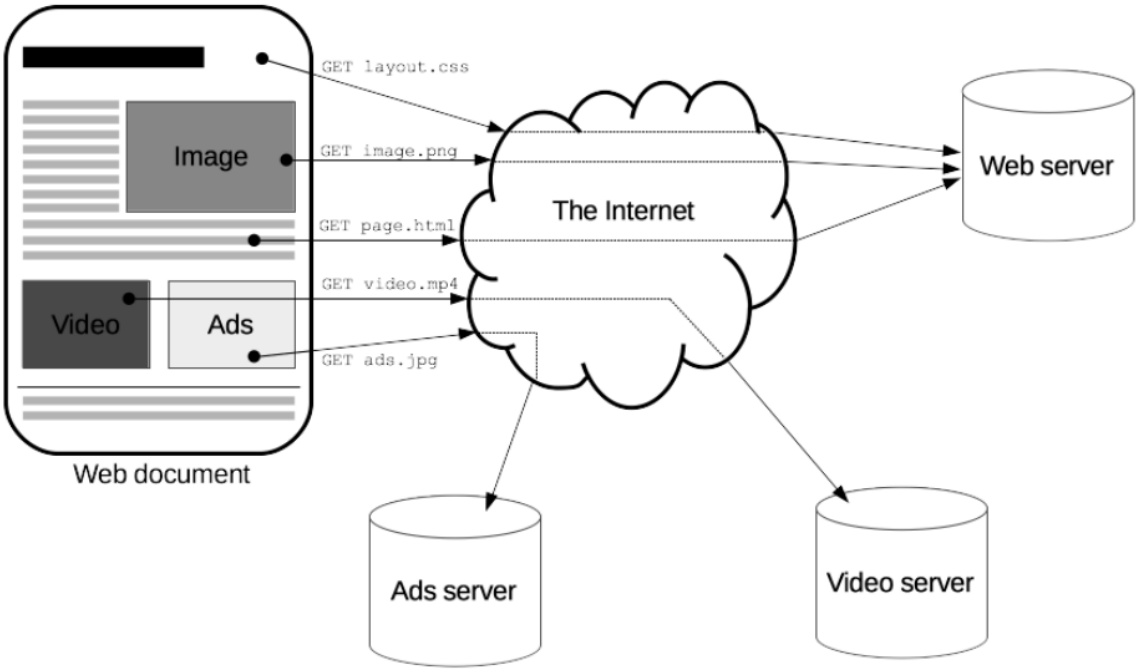
Developer.mozilla.org. 2005. *Generalidades del protocolo HTTP - HTTP | MDN*. [online] Available at: <https://developer.mozilla.org/es/docs/Web/HTTP/Overview> [Accessed 3 May 2021].

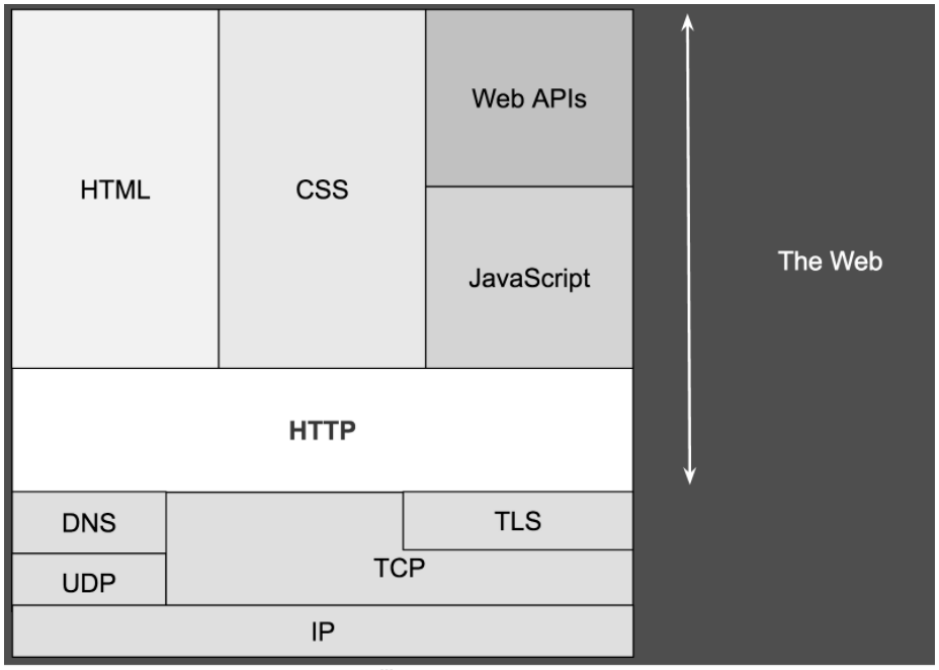
Delgado B., M., 2016. *5.9 Servidor Web*. [online] Upanama.e-ducativa.com. Available at: <https://upanama.e-ducativa.com/archivos/repositorio/6000/6126/html/59_servi.htm> [Accessed 2 May 2021].

Morero, F., 2000. *Introducción a la OOP*. [online] Kataix.umag.cl. Available at: <https://kataix.umag.cl/~ruribe/Utilidades/Introduccion%20a%20la%20Programacion%20Orientada%20a%20Objetos.pdf> [Accessed 5 March 2021].

ANEXOS

VIII. PROTOCOLO HTTP





IX. ARQUITECTURA CLIENTE - SERVIDOR